RSAC 2025 Conference

Many Voices.
One Community.

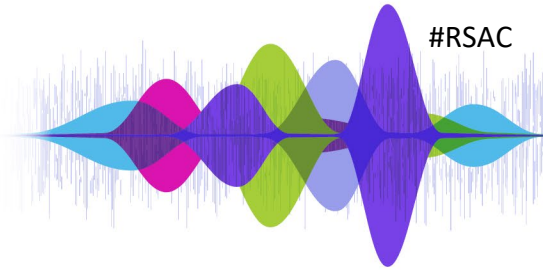
SESSION ID: CRYP-M06A

Affine Equivalence-Based Key-Recovery Attacks on White-Box Implementations of the SM4 Block Cipher

Jiqiang Lu

Professor
Beihang University

Disclaimer



Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA Conference LLC or any other co-sponsors. RSA Conference LLC does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

© 2025 RSA Conference LLC or its affiliates. The RSAC and RSAC CONFERENCE logos and other trademarks are proprietary. All rights reserved.

Affine Equivalence-Based Key-Recovery Attacks on White-Box Implementations of the SM4 Block Cipher

Zexuan Chen, Jiqiang Lu

School of Cyber Science and Technology,
Beihang University
HangZhou Innovation Institute,
Beihang University
czxyeah, lvjiqiang@buaa.edu.cn

CT-RSA 2025

Outline

1. Preliminaries
2. Affine Equivalence-Based Key-Recovery Attack Framework
3. Key-Recovery Attack on Xiao and Lai's Implementation
4. Key-Recovery Attack on Bai and Wu's Implementation
5. Key-Recovery Attack on Shi et al.'s Implementation
6. Summary

1.1 SM4 Block Cipher

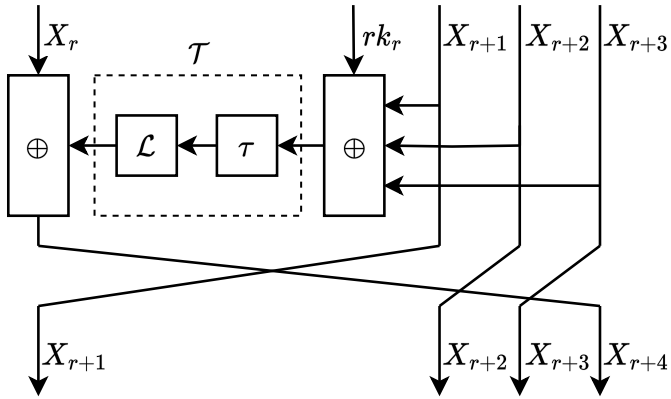
- The SM4 block cipher, previously designated as SMS4, was elevated to a Chinese national standard in 2016 before achieving ISO international standardization in 2021.
- A generalised Feistel structure with a 128-bit block size and four 32-bit branches, featuring a 128-bit user key and a total of 32 rounds.
- Round Function:
  - Denote by $\mathcal{F}$ the round encryption function, by $(X_r, X_{r+1}, X_{r+2}, X_{r+3})$ the 128-bit input to the r-th round, and by rk_r the 32-bit r-th round key, where $X_r \in GF(2)^{32}$ and $0 \leq r \leq 31$. The r-th round encryption of SM4 is given by

$$X_{r+4} = \mathcal{F}(X_r, X_{r+1}, X_{r+2}, X_{r+3}, rk_r) = X_r \oplus \mathcal{T}(X_{r+1}, X_{r+2}, X_{r+3}, rk_r),$$

    where $\mathcal{T} : GF(2)^{32} \to GF(2)^{32}$ is composed of a non-linear transformation τ and a linear transformation $\mathcal{L}$, that is, $\mathcal{T}(x) = \mathcal{L}(\tau(x))$.
  - Non-linear transformation τ: mapping $GF(2)^{32} \to GF(2)^{32}$ is composed of four parallel 8-bit S-boxes and is defined as $\tau(x) = (S(x_0)||S(x_1)||S(x_2)||S(x_3))$;
  - Linear transformation $\mathcal{L}$: mapping $GF(2)^{32} \to GF(2)^{32}$ is defined as $\mathcal{L}(x) = x \oplus (x \lll 2) \oplus (x \lll 10) \oplus (x \lll 18) \oplus (x \lll 24)$. $\mathcal{L}$ can also be represented as a 32×32-bit non-singular matrix $\begin{bmatrix} B_1 & B_2 & B_2 & B_3 \\ B_3 & B_1 & B_2 & B_2 \\ B_2 & B_3 & B_1 & B_2 \\ B_2 & B_2 & B_3 & B_1 \end{bmatrix}$, where B_1, B_2 and B_3 are all 8×8-bit non-singular matrix.

1.1 SM4 Block Cipher

- The r-th round encryption of SM4 is as follows.



- Key Schedule:
  - For $0 \leq r \leq 31$, the round keys are generated from a 128-bit user key $MK = (MK_0, MK_1, MK_2, MK_3) \in (GF(2)^{32})^4$ in this way:

$$(K_0, K_1, K_2, K_3) = (MK_0 \oplus FK_0, MK_1 \oplus FK_1, MK_2 \oplus FK_2, MK_3 \oplus FK_3),$$
$$rk_r = K_{r+4} = K_r \oplus \widehat{\mathcal{T}}(K_{r+1} \oplus K_{r+2} \oplus K_{r+3} \oplus CK_r),$$

where FK_j $(j = 0, 1, 2, 3)$ and CK_r are 32-bit public constants, and $\widehat{\mathcal{T}}$ is the linear transformation of $\mathcal{T}$ with $\mathcal{L}$ being replaced by the linear transformation $\widehat{\mathcal{L}} = x \oplus (x \lll 13) \oplus (x \lll 23)$.

1.2 White-Box Cryptography and White-Box SM4 Implementation

- In 2002, Chow et al. introduced white-box cryptography, focusing on security in a white-box model where attackers fully control the execution environment, and proposed a white-box AES implementation using networked lookup tables.
- The design methods of current white-box SM4 implementations are inspired by those of white-box AES implementations:
  - **Xiao and Lai:** First white-box SM4 using affine encodings in 2009;
  - **Shi et al.:** Lightweight white-box SM4 with linear encodings in 2015;
  - **Bai and Wu:** White-box SM4 using shared data with affine encodings in 2016;
  - **Lin et al.:** Key-recovery attack (2^{49} complexity) on Shi's implementation in 2018;
  - **Pan et al.:** Security analysis of previous SM4 implementations in 2018;
  - **Yao and Chen:** New white-box SM4 with dummy states and affine encodings in 2020;
  - **Lu et al.:** Proved collision-based attacks ineffective on white-box SM4 implementations in 2023.
- The difference between SM4's generalized Feistel structure and AES's SPN structure is a key reason why no known key-recovery attacks exist for current white-box SM4 implementations using affine transformations.

1.3 Derbez et al.'s Affine Equivalence Result

- Affine Equivalence: Two known n-bit non-linear bijections $\mathcal{F}$ and $\mathcal{S}$ are defined to be affine equivalent if there exist two n-bit affine mappings $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{F} = \mathcal{B} \circ \mathcal{S} \circ \mathcal{A}$.
- Self-Equivalence: An n-bit non-linear bijection $\mathcal{S}$ is defined to be self-equivalent if there exists more than one pair of n-bit affine mappings $(\mathcal{A}, \mathcal{B})$ such that $\mathcal{S} = \mathcal{B} \circ \mathcal{S} \circ \mathcal{A}$.

- Derbez et al.'s Affine Equivalence Algorithm:

  Knowing $\mathcal{S}_i$ and given oracle access to $\mathcal{F}$ (but not $\mathcal{F}^{-1}$), the algorithm finds two affine mappings $\mathcal{A}'$ and $\mathcal{B}'$ satisfying $\mathcal{F} = \mathcal{B}' \circ (\mathcal{S}_1, \cdots, \mathcal{S}_k) \circ \mathcal{A}'$ with time complexity:For k identical non-self-equivalent S-boxes:

$$O(2^m \cdot n^3 + l \cdot 2^m \cdot n^3 + k \cdot n^3 + k \cdot m^3 \cdot 2^m)$$

  For k identical self-equivalent S-boxes:

$$O(2^m \cdot n^3 + l \cdot 2^m \cdot n^3 + k \cdot n^3 + k \cdot m^3 \cdot 2^{2m})$$

  For k different non-self-equivalent S-boxes:

$$O(l \cdot 2^m \cdot n^3 + k \cdot n^3 + m \cdot n^2 \cdot 2^m)$$

  Otherwise:

$$O(l \cdot 2^m \cdot n^3 + k \cdot n^3 + m \cdot n^2 \cdot 2^{2m})$$

  where l denotes the number of test values.

2.1 SM4 S-box Self-Equivalence

- Expression of the 2040 Pairs of Affine Mappings for SM4 S-box Self-Equivalence
  - The SM4 S-box is defined as

$$S(x) = \mathcal{M}((\mathcal{M} \cdot x)^{-1}),$$

where $\mathcal{M}(\cdot)$ is a fixed affine mapping over a finite field and $x \in GF(2^8)$.

  - Let $[a]$ be the 8×8-bit matrix that corresponds to a multiplication by a in $GF(2^8)$ and $\mathcal{H}$ be the 8×8-bit matrix that performs the squaring operation in $GF(2^8)$. A general expression for 2040 pairs of affine mapping $\mathcal{A}$ and $\mathcal{B}$ that satisfy $S = \mathcal{B} \circ S \circ \mathcal{A}$ can be derive:

$$\mathcal{A}(x) = \mathcal{M}^{-1}\left([a] \cdot \mathcal{H}^i \cdot \mathcal{M} \cdot x\right),$$
$$\mathcal{B}(x) = \mathcal{M}\left(\mathcal{H}^{-i} \cdot [a] \cdot \mathcal{M}^{-1} \cdot x\right),$$

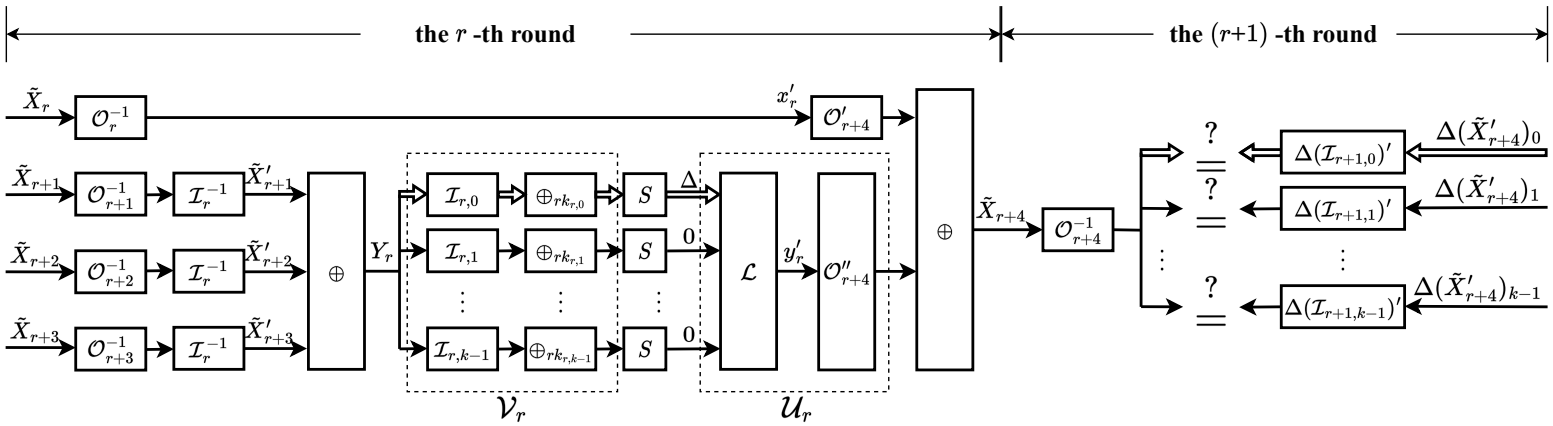
where $0 \leq i \leq 7$ and $a \in GF(2^8)\backslash\{0\}$.

2.2 Our Attack Framework

- White-box SM4 Implementation Structure:
  - Uses affine/linear encodings to protect branches and round functions;
  - Generates k white-box tables ($m \times n$-bit) per round;
  - Employs diagonal input encoding $\mathcal{I}_r$ (k $m \times m$-bit affine/linear encodings);
  - Uses $n \times n$-bit affine/linear output encoding $\mathcal{O}''_{r+4}$.
- Representation of White-Box Tables:

$$F_r = \mathcal{U}_r \circ (S, \cdots, S) \circ \mathcal{V}_r,$$

  - $\mathcal{V}_r = \oplus_{rk_r} \circ \mathcal{I}_r$ (input transformations), $\mathcal{U}r = \mathcal{O}''_{r+4} \circ \mathcal{L}$ (output transformations);
  - Parameters: $n = m \cdot k$, with 4 parallel 8-bit SM4 S-boxes per round.
- All known white-box SM4 implementations fit this format and are vulnerable to our three-phase attack process as follows:



2.2 Our Attack Framework

- Phase 1 - Obtaining All the Equivalent Affine Encodings
  - Goal: Find all the pairs of equivalent affine mappings $\mathcal{V}'_r$ and $\mathcal{U}'_r$ such that $F_r = \mathcal{U}'_r \circ (S, \cdots, S) \circ \mathcal{V}'_r$.
  - Key Steps:
    - Apply Derbez's algorithm to compute input difference spaces $I_{r,i}$ for each S-box and form $I_r = diag(I_{r,0}, \cdots, I_{r,k-1})$;
    - Due to S-box's self-equivalence property, the computed I_r may not equal $\mathcal{V}_r$; and there are 2040 equivalent representations for $\mathcal{V}'_{r,i}$ that correspond to the correct affine mapping $\mathcal{V}_{r,i}$ (that makes the i-th S-box active), forming the diagonal structure $\mathcal{V}'_r = diag(\mathcal{V}'_{r,0}, \cdots, \mathcal{V}'_{r,k-1})$;
    - Corresponding $\mathcal{U}'_r$ is determined for each $\mathcal{V}'_r$.
  - Time Complexity: $O(k \cdot m^3 \cdot 2^{2m})$

2.2 Our Attack Framework

- Phase 2 - Identifying the Correct Affine Encodings
  - Goal:
    - Identify the correct affine encodings $\mathcal{V}_r$ and $\mathcal{U}_r$;
    - Handle challenges caused by Feistel structure, where output encodings $\mathcal{O}'_{r+4}$ and $\mathcal{O}''_{r+4}$ are linearly related but include a secret constant.
  - Key Steps:
    - Use a differential meet-in-the-middle approach over two consecutive rounds;
    - Set Y_r so only the i-th S-box in round r outputs Δ, resulting in $\mathcal{L} \cdot (0, \ldots, 0, \Delta, 0, \ldots, 0)$;
    - Set $\tilde{X}_r = 0$ and use the same Y_r's to compute k m-bit differences $\Delta(\tilde{X}'_{r+4})_i$ in round $(r+1)$;
    - Since the encoding $\mathcal{O}_{r+4}$ will be canceled, applying the linear part (denoted by $\Delta(\mathcal{I}_{r+1,i})$) of the correct one out of the 2040 candidates $\mathcal{V}'_{r+1,i}$ to $\Delta(\tilde{X}'_{r+4})_i$ will equal $(\mathcal{L} \cdot (0, \ldots, 0, \Delta, 0, \ldots, 0))_i$.
  - Time Complexity: $O(2k \cdot 2040^t)$, where t is a small integer (e.g., 1 or 2).

2.2 Our Attack Framework

- Phase 3 - Recovering a Round Key as well as the User Key
  - Goal: Recover both round key and user key by analyzing the known encodings.
  - Key Steps:
    - For linear encoding:
      - Extract the round key directly as $\mathcal{V}_r(0)$;
      - Use four recovered round keys to deduce the user key.
    - For affine encoding:
      - Recover the linear relationship between the round key and unknown vectors in $\mathcal{I}_r$ as $\mathcal{V}_r(0)$;
      - Analyze three additional consecutive rounds to recover more white-box parameters;
      - Use SM4's key expansion formula to deduce a 32-bit unknown;
      - Solve the 32-bit unknown (round key or white-box parameter) via exhaustive search to recover the full key.
  - Time Complexity: $O(t^2 \cdot 2^{32})$ for the case of affine encodings.

2.2 Our Attack Framework

- Total time complexity:
  - For linear encoding:

$$O(t \cdot k \cdot m^3 \cdot 2^{2m} + 2k \cdot 2040^t + t^2 \cdot 2^{32})$$

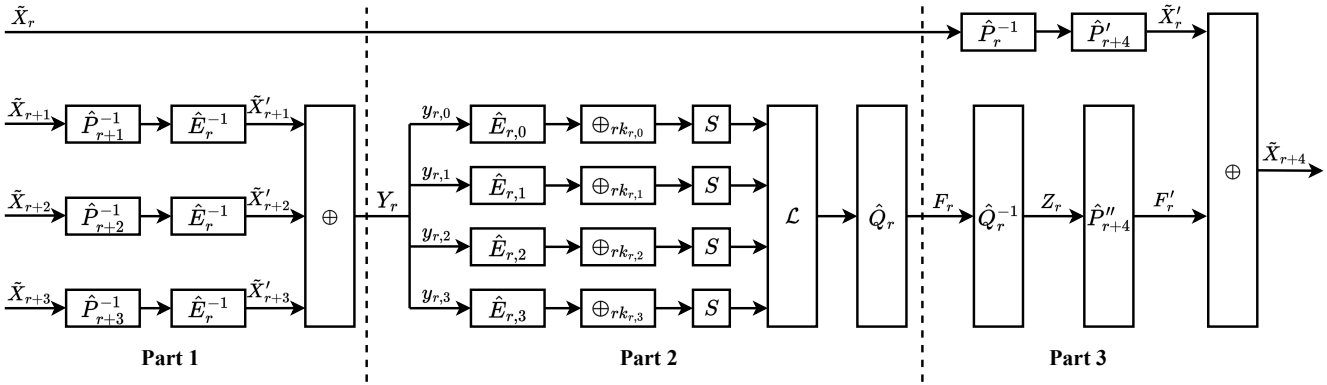
  - For affine encoding:

$$O(t \cdot k \cdot m^3 \cdot 2^{2m} + 2k \cdot 2040^t)$$

- Remark:
  - White-box SM4 implementations do not use non-linear encodings;
  - If non-linear encodings were used, they could be removed first using existing methods.
    - Baek, C.H., Cheon, J.H., Hong, H.: White-box AES implementation revisited. Journal of Communications and Networks, Vol. 18(3), 273–287 (2016).
    - Billet, O., Gilbert, H., Ech-Chatbi C.: Cryptanalysis of a White Box AES Implementation. In: SAC 2004, LNCS 3357, pp. 227–240. Springer (2004).

3.1 Xiao and Lai's Implementation

- The implementation of an encryption round consists of three parts as follows.



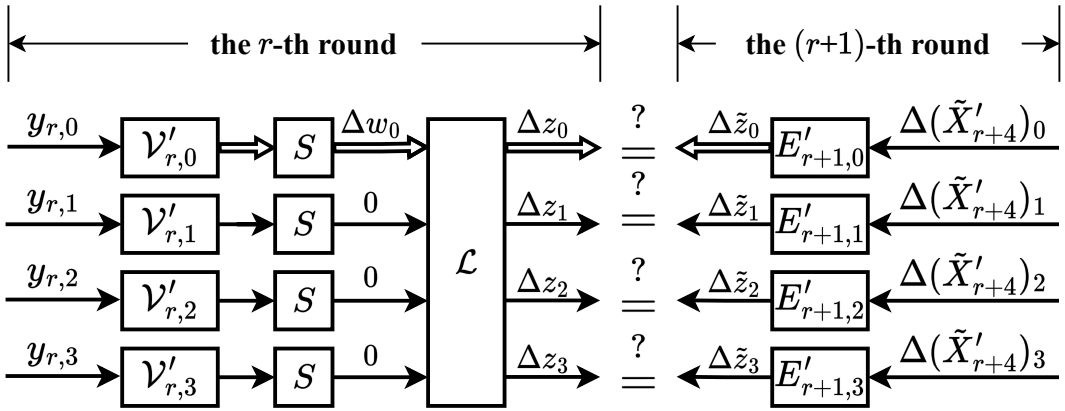
- Part 1 - Implementation of $\tilde{X}_{r+1} \oplus \tilde{X}_{r+2} \oplus \tilde{X}_{r+3}$.
  - For $0 \leq r \leq 31$, the original values $\tilde{X}_{r+j}$ $(j = 1, 2, 3)$ are recovered by canceling $\hat{P}_{r+j}$ and applying $\hat{E}_r^{-1}$, expressed as $\tilde{X}'_{r+j} = \hat{E}_r^{-1} \circ \hat{P}_{r+j}^{-1}(\tilde{X}_{r+j}) = \hat{M}_{r+j}^{r}(\tilde{X}_{r+j})$.
- Part 2 - Implementation of $\mathcal{T}(Y_r \oplus rk_r)$.
  - Four 8-to-32-bit lookup tables are built using input/output affine encodings, S-boxes, key addition, and linear transformations, with the output as $F_r = \mathcal{U}_r \circ (S, \cdots, S) \circ \mathcal{V}_r$.
- Part 3 - Implementation of $\tilde{X}_{r+4}$.
  - The value $\tilde{X}_{r+4}$ is computed from F_r and $\tilde{X}_r$ using affine mappings and XOR as $\tilde{X}_{r+4} = \tilde{X}'_r \oplus F'_r$.

3.2 Attacking Xiao and Lai's Implementation

- Phase 1 - An Equivalent Representation of Lookup
  - Goal: Find equivalent affine mappings $\mathcal{U}'_r$ and $\mathcal{V}'_r$ such that: $F_r = \mathcal{U}'_r \circ (S, \cdots, S) \circ \mathcal{V}_r$.
  - Computing $\mathcal{V}'_r$:
    - Create input form: $\oplus_{i=0}^{3} Table_i(0, \cdots, 0, y_j, 0, \cdots, 0)$;
    - Build an 8-bit to 32-bit mapping: $G \circ S \circ \mathcal{V}_i$, and compute a projection P such that $P \circ G \circ S \circ \mathcal{V}_i$ becomes a bijection that is affine-equivalent to the SM4 S-box.
    - Apply the affine equivalence algorithm to find $\mathcal{V}_i$. Due to the 2040 self-equivalence relations of the SM4 S-box, the algorithm may not recover $\mathcal{V}_i$ exactly. However, there exists the following relation: $\mathcal{V}'_i = \mathcal{A} \circ \mathcal{V}_i$, where $\mathcal{A}$ has 2040 possible values.
  - Recovering $\mathcal{U}'$:
    - Split the linear part U' of $\mathcal{U}'$ into components $U' = (U'_0, \cdots, U'_3)$, and use any candidate for each $\mathcal{V}'_i$ to compute each U'_i using the canonical basis vectors;
    - Recover the full affine mapping $\mathcal{U}'$ by combining U' with trhe vector part u'.
  - Time Complexity: $O(4 \times 8^3 \times 2^{2\times 8}) = O(2^{27})$, dominated by computing four 8-bit affine transformations $\mathcal{V}'_i$.

3.2 Attacking Xiao and Lai's Implementation

- Phase 2 - Identifying the Correct Encodings
  - Goal: To recover the correct encodings from two consecutive rounds.
  - Key Steps:
    - For each $\mathcal{V}_{r,0}$ candidate, compute differential values Δz_0^l using $\mathcal{L} \cdot (\Delta w_0^l, 0, 0, 0)$ and store them in a hash table indexed by Δz_0^l;
    - For each $\mathcal{V}r+1,0$ candidate, compute $\Delta \tilde{z}0^l$ and use a meet-in-the-middle approach to check for matches in the hash table. If a match is found, identify the correct $\mathcal{V}r,0$ and $\mathcal{V}r+1,0$;
    - Apply the same method to recover $\mathcal{V}r,j$ and $\mathcal{V}r+1,j$ for $j=1,2,3$.
  - Time Complexity : $O(8 \cdot 2040) \approx O(2^{14})$, dominated by building and matching hash tables for 2040 candidates per block.



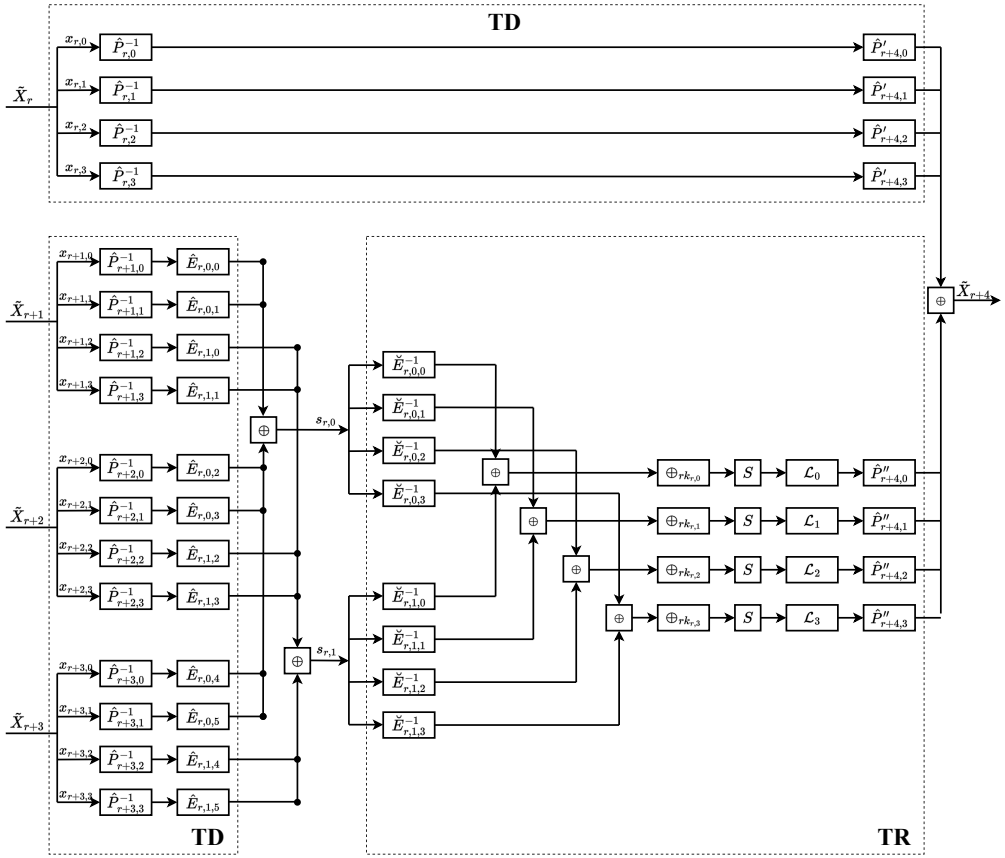
3.2 Attacking Xiao and Lai's Implementation

- Phase 3 - Recovering the User Key
  - Goal: To recover the round keys and external encodings using the identified correct encodings.
  - Key Steps:
    - Recover invertible matrices and compute XOR operations between two consecutive round keys by using public affine mappings $\hat{M}_{r+3}^{r+1}$;
    - Repeating all the previous steps of the attack for a few subsequent rounds, the value of an XOR operation performed between any two round keys can be computed;
    - Recover rk_r with a brute-force search of 2^{32} possibilities using the SM4 key expansion algorithm.
  - Time Complexity : $O(2^{32})$, dominated by exploring all possibilities of round key rk_r.

3.2 Attacking Xiao and Lai's Implementation

- Summary:
  - Feistel Structure Insight:
    - One branch ($\tilde{X}_{r+3}$) entering a round function also enters the next round function;
    - The white-box protection form (P_{r+3}^{-1}) of this branch remains unchanged across two rounds.
  - Key Recovery:
    - Use recovered white-box operations to compute $e_r \oplus e_{r+1}$ and the XOR of two round keys;
    - By analyzing subsequent rounds, derive the XOR values of any two consecutive round keys;
    - Represent all round keys using one round key via SM4's key expansion formula.
  - Implementation Results:
    - we implemented this attack in `C++` on a desktop computer with a single core (`Intel Core i7-7700 CPU @ 3.60GHz`), and it recovered the 128-bit user key and the external input and output encodings in less than two minutes.

4.1 Bai and Wu's Implementation



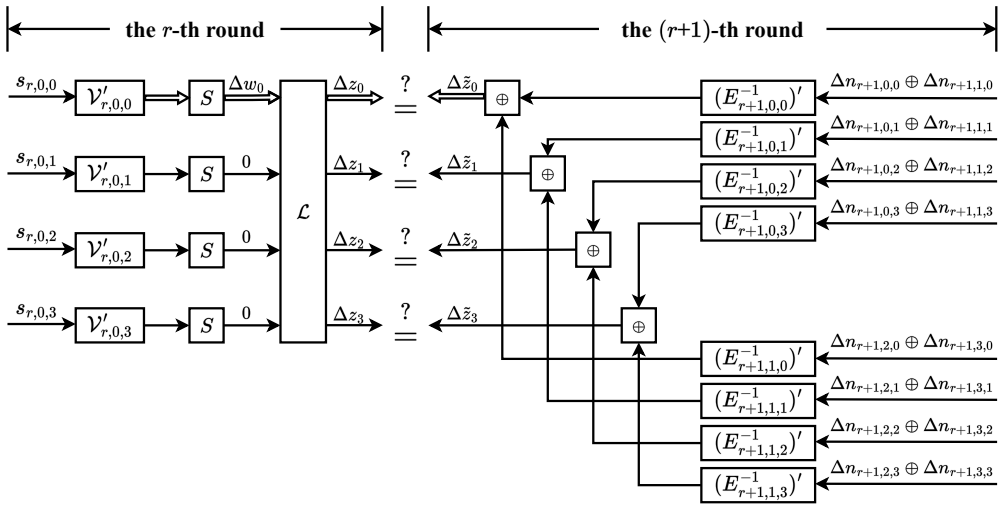
- The implementation uses only affine encodings, and consists of two types of lookup tables.
- Sixteen 8-bit to 32-bit $\mathbf{TD}_{r,i,j}$ tables $(0 \leq r \leq 31, 0 \leq i, j \leq 3)$;
- Four 16-bit to 32-bit $\mathbf{TR}_{r,j}$ tables $(0 \leq r \leq 31, 0 \leq j \leq 3)$;
- 32-bit XOR operations on 4 **TD** tables and 4 **TR** tables to produce the 32-bit output $\tilde{X}_{r+4}$.

4.2 Attacking Bai and Wu's Implementation

- Phase 1 - An Equivalent Representation of Lookup Tables
  - Preprocessing:
    - Convert four 16-bit to 32-bit **TR** tables into eight 8-bit to 16-bit **TR** tables by considering $\mathbf{TR}_{r,j}(\alpha, 0)$ and $\mathbf{TR}_{r,j}(0, \beta)$ with $\alpha, \beta \in GF(2)^8$;
    - Represent the tables as eight 8-bit to 32-bit lookup tables $F_{r,h,j} = \mathcal{U}_{r,j} \circ S \circ \mathcal{V}_{r,h,j}$ and there exists $F_{r,h} = \oplus_{j=0}^{3}(\mathcal{U}_{r,j} \circ S \circ \mathcal{V}_{r,h,j}) = \mathcal{U}_r \circ (S, \cdots, S) \circ \mathcal{V}_{r,h}$.
  - Computing $\mathcal{V}'_h$:
    - Since $\mathcal{V}_h$ is block diagonal with 8-bit blocks, only one S-box is active for specific inputs;
    - There are 2040 candidates to find for each block $\mathcal{V}'_{h,j}$.
  - Recovering of $\mathcal{U}'$:
    - Once $\mathcal{V}'_{h,j}$ is recovered, $\mathcal{U}'$ can be derived easily.
    - Use the relationship $F_{r,h} = \mathcal{U}_r \circ (S, \cdots, S) \circ \mathcal{V}_{r,h}$ to compute $\mathcal{U}'$.
  - Time Complexity: $O(2^{28})$, as 2040 candidates must be explored for each of the eight $\mathcal{V}'_{h,j}$ blocks, with 2^{16} input-output pairs per block.

4.2 Attacking Bai and Wu's Implementation

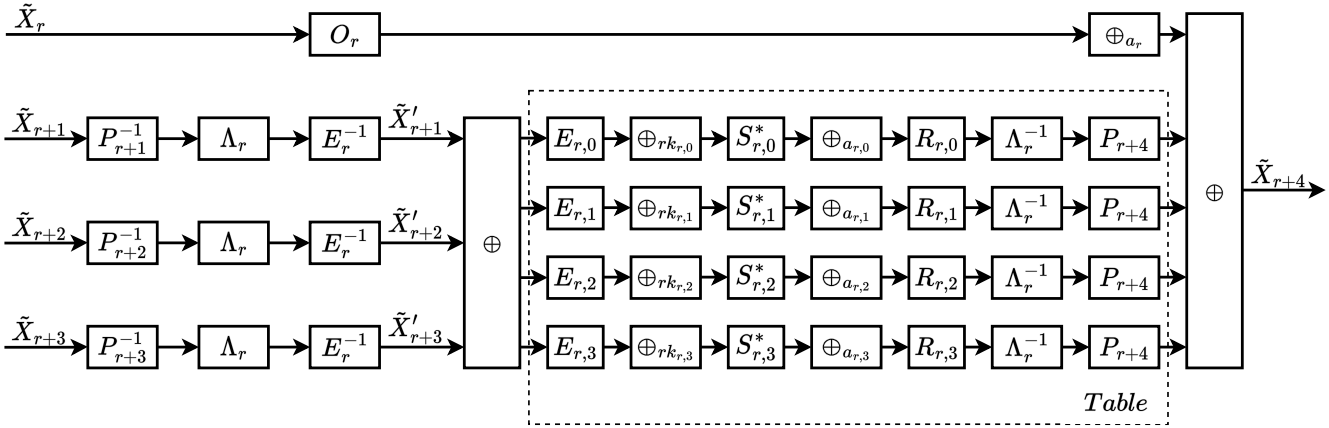
- Phase 2 - Identifying the Correct Encodings
  - Goal:Recover the correct affine mappings $\mathcal{V}_{r,h}$.
  - Key Steps:
    - Analyze two consecutive rounds to derive the relationships between affine mappings;
    - Construct specific input differences to eliminate fixed vectors in the affine mappings;
    - Use a meet-in-the-middle approach to recover $\mathcal{V}_{r,h,j}$ and $\mathcal{V}_{r+1,h,j}$ for all blocks.
  - Time Complexity: $O(2^{25})$, primarily due to hash table operations and recovering affine mappings from 2040 candidates.



4.2 Attacking Bai and Wu's Implementation

- Phase 3 - Recovering the User Key
  - Goal:Recover all round keys and external encodings.
  - Key Steps:
    - Use the recovered affine encodings $\mathcal{V}r, h, j$ and $\mathcal{V}r+1, h, j$ to compute matrices $E_{r,0}$, $E_{r,1}$, $E_{r+1,0}$, $E_{r+1,1}$ and vectors v_r, v_{r+1};
    - Set specific inputs to compute the XOR relations between any two rounds of the round keys and their corresponding vectors: $P_r^{-1} \cdot p_r \oplus P_{r+i}^{-1} \cdot p_{r+i} \oplus rk_r \oplus rk_{r+i} = \gamma_{r,i}$;
    - Use the XOR operation and the SM4's key expansion formula to recover to the 32-bit value of $P_r^{-1} \cdot p_r$. Since $t_{r-1} = P_r^{-1} \cdot p_r \oplus P_{r+1}^{-1} \cdot p_{r+1} \oplus P_{r+2}^{-1} \cdot p_{r+2} \oplus rk_{r-1}$ can be easily obtained, the value of $P_{r+j}^{-1} \cdot p_{r+j} (j = 0, 1, 2)$ needs to be computed separately three times;
    - The round key rk_{r-1} can be recovered, but in order to recover the external encoding, we need to search again up to 2^{32} times to get the value of $P_{r-1}^{-1} \cdot p_{r-1}$.
  - Time Complexity: $O(2^{34})$, dominated by four 32-bit searches $(4 \cdot 2^{32})$ to recover round keys and external encodings.

5. Key-Recovery Attack on Shi et al.'s Implementation



- This implementation is similar to Xiao and Lai's but uses linear encodings.
  - Key Steps:
    1. Use the equivalent lookup table representation to recover the 32-bit affine encodings $\mathcal{V}_r$ and $\mathcal{U}_r$ through a differential and meet-in-the-middle approach;
    2. No further action is needed after recovering $\mathcal{V}_r$, as its vector part directly reveals the round key rk_r;
    3. External encodings can be recovered with complexity $O(1)$, as the encodings are linear.
  - Time Complexity: $O(2^{27})$, dominated by recovering $\mathcal{V}'_{r,j}$ $(j = 0, 1, 2, 3)$.

6. Summary

- Proposed Framework: Developed an affine equivalence-based attack framework to recover the original user key from white-box SM4 implementations with affine (or linear) encodings.
- Applications:
  - Successfully applied the framework to:
    - Xiao and Lai's implementation: 2^{32} complexity;
    - Bai and Wu's implementation: 2^{34} complexity;
    - Shi et al.'s implementation: 2^{27} complexity;
    - Yao and Chen's implementation can also be attacked with 2^{32} complexity by reducing it to Xiao and Lai's implementation;
    - In addition to the above implementations, the framework can also attack other existing white-box SM4 implementations.
- Key Insights:
  - Demonstrated how Derbez et al.'s affine equivalence algorithm for SPN ciphers can be adapted to generalized Feistel ciphers;
  - Highlighted that existing white-box SM4 designs are not practically secure.

RSAC™ 2025 Conference

Many Voices.
One Community.

Thank You!